#14/B

PATENT

Attorney Docket No. 02307I-553

BH 6-19-97

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	
MICHAEL D. DOYLE et al.	Examiner: D. Dinh
Application No.: 08/324,443	Art Unit: 2317
Filed: 10/17/94	AMENDMENT
For: EMBEDDED PROGRAM OBJECTS IN DISTRIBUTED HYPERMEDIA SYSTEMS	

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Responsive to the Office Action mailed March 26, 1997, please amend the above identified application as follows:

IN THE CLAIMS:

Please cancel claim 6-15, 17-43, and 49-56.

Please amend the following claims:

1. (Twice Amended) A method for running an application program in a computer network environment, comprising:

providing at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment;

executing, at said client workstation, a browser application, that parses a first distributed hypermedia document to identify text formats included in [the] said distributed hypermedia document and for responding to predetermined text formats to initiate processing specified by said text formats; utilizing said browser to display, on said client workstation, at

least a portion of a first hypermedia document received over said network from said server, wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation, [and] wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document[and that specifies], wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document[;], and wherein said embed text format is parsed by said browser to automatically invoke [invoking, with said browser application,] said executable application to execute on said client workstation in order to display said object and enable interactive processing of said object within [the] a display ~~window~~ area created at said first location within the portion of said first distributed hypermedia document being displayed in said [the] first browser-controlled window [while a portion of said first distributed hypermedia document continues to be displayed within said browser-controlled window].

2. (Twice Amended) The method of claim 1, wherein said executable application is a controllable application and further comprising the step of:

interactively controlling said controllable application (from] on said client workstation via [communications sent over said distributed hypermedia environment] inter-process communications between said browser and said controllable application.

4 ~~3~~. (Twice Amended) The method of claim ~~2~~ 3 [2], wherein additional instructions for controlling said controllable application reside on said network server, wherein said step of

interactively controlling said controllable application includes the following substeps:

issuing, from the client workstation, one or more commands to the network server;

executing, on the network server, one or more instructions in response to said commands;

sending information from said network server to said client workstation in response to said executed instructions; and

processing said information at the client workstation to interactively control said controllable application.

5~~4~~. (Twice Amended) The method of claim 4~~2~~ [2], wherein said additional instructions for controlling said controllable application reside on said client workstation.

3~~5~~. (Twice Amended) The method of claim 2, wherein the communications to interactively control said controllable application [from said client workstation] continue to be exchanged between the controllable application and the [hypermedia] browser even after the controllable application program has been launched.

6~~44~~. (Amended) A computer program product for use in a system having at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment, the computer program product comprising:

a computer usable medium having computer readable program code physically embodied therein [for causing a client workstation to invoke an external executable application referenced by a hypermedia document to display and process an external object referenced by the hypermedia document], said computer program product further comprising:

computer readable program code for causing said client workstation to execute a browser application to parse a first distributed hypermedia document to identify text formats included in [the] said distributed hypermedia

B1

33

document and to respond to predetermined text formats to initiate processes specified by said ~~by the~~ text formats;

computer readable program code for causing said client workstation to utilize said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation, [and] wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document[and that specifies], wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document[;

computer readable program code for causing said client workstation to invoke, with said browser application], and wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable interactive processing of said object within [the] a display ~~window~~ area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window [while a portion of said first distributed hypermedia document continues to be displayed within said browser-controlled window].

45. (Amended) The computer program product of claim 44, wherein said executable application is a controllable application and further comprising:

computer readable program code for causing said client workstation to interactively control said controllable application [from] on said client workstation via [conununications sent over said distributed hypermedia

environment] inter-process communications between said browser and said controllable application.

9 ~~46~~. (Amended) The computer program product of claim ~~48~~ 8 [45], wherein additional instructions for controlling said controllable application reside on said network server, wherein said step of interactively controlling said controllable application includes:

computer readable program code for causing said client workstation to issue, from the client workstation, one or more commands to the network server;

computer readable program code for causing said network server to execute one or more instructions in response to said commands;

computer readable program code for causing said network sever to send information to said client workstation in response to said executed instructions; and

computer readable program code for causing said client workstation to process said information at the client workstation to interactively control said controllable application.

10 ~~47~~. (Amended) The computer program product of claim ~~46~~ 9 [45], wherein said additional instructions for controlling said controllable application reside on said client workstation.

7 8 ~~48~~. (Amended) The computer program product of claim ~~45~~, wherein the communications to interactively control said controllable application [from said client workstation] continue to be exchanged between the controllable application and the [hypermedia] browser even after the controllable application program has been launched.

REMARKS

Claims 1-15 and 17-56 have been examined, claims 1-5 and 44-48 are amended herein, and claims 6-15, 17 43, and 49-56 are canceled. Accordingly, claims 1-5 and 44-48 are now pending in the application.

35

THE REJECTION OF CLAIM 1

Claim 1 is rejected under 35 U.S.C. §103 as being unpatentable over Applicants' disclosed prior art (Mosaic, HTTP, HTML, and the World-Wide Web) and further in view of Khoyi et al. and Hansen.

THE CLAIMED INVENTION

The present invention, as defined for example in amended claim 1, includes the step of executing a browser that parses a first distributed hypermedia document to identify text formats included in the distributed hypermedia document and that responds to predetermined text formats to initiate processing specified by the text formats. The browser displays a portion of a first distributed hypermedia document in a browser-controlled window.

The first distributed hypermedia document includes an embed text format located at a first location in the document. The embed text format specifies the location of an object, at least a portion of which is external to the first distributed hypermedia document, that has type information associated with it which is utilized by the browser to identify and locate an executable application external the document.

The embed text format is parsed by the browser to cause the browser to automatically invoke the external application to execute on the client workstation. The external application displays, and allows the user to interactively process, the object in a display window created within the portion of the document being displayed in the browser-controlled window, at the location within the document of the embed text format.

THE CITED REFERENCES

1. Mosaic.

The Applicants' prior art (Mosaic) launches helper applications, in response to a user's interactive command, in a separate window to view certain types of file types. As described in the specification, the mechanism for specifying and locating a linked object is an HTML anchor "element" that

includes an object address in the format of Uniform Resource Locator (URL) (pg. 3, line 30). Many viewers exist that handle various file formats such as ".TIF", .GIF"., etc. When a user commands the browser program to invoke a viewer program, typically by clicking on an anchor with a mouse, the viewer is launched as a separate process. The viewer displays the full image in a separate "window" (in a windowing environment) or on a separate screen. This means that the browser program is no longer active while the viewer is active. The viewer program is completely independent of the browser after being invoked by the browser. This means that there is no communication between the viewer program and the browser program after the viewer program has been launched. As a result, the viewer program continues to run, even after the browser program execution is stopped, unless the user explicitly stops the viwer program's execution.

The attached pages (attachment A) describe how helper applications are invoked in Netscape Navigator, which uses the same mechanism as Mosaic. The user creates an association in a table between a file extension, e.g., the file extension ".MPG" indicates a file formatted in MPEG video. The browser could be configured to launch the helper application VMPEG to display a video file accessed using a URL in hypermedia document. As described above, the MPEG video file would be displayed in a separate window and the browser would be inactive.

2. Khoyi et al.

The reference Khoyi et al. describes an object-based data processing operating system. In that operating system data from a child object may be internalized in a parent object. Objects are related to one another through a linking mechanism (col. 10, line 21). The terms "parent" and "child" refer to the direction of the link (col. 10, line 31). Programs for operating on objects are known as "object managers", sometimes referred to as applications (col. 9, line 17). Each type of object has associated with it at least one object manager designed as the primary means for operating on data stored in that type of object (col. 9, line 20). For example, the system

may support a "document type" of object for word processing and a word processing object manager will be associated with that object type. Similarly, a "data base type" object will have associated with it a data base object manager as the primary means for operating upon data stored in the data base type object.

Application Integration Services of this operating system provide a mechanism by which an individual application (object manager) can appear to a user to integrate its operation and manipulation of data with that of other applications (col. 13, line 40). To a user, the display of a page of a newsletter having both text and a picture indicates to the user that the text and picture are integrated into a single document. However, in the Khoyi system this integration effect is accomplished by the operation of two different object managers coordinated by the use of the operating system's Application Integration Services. The newsletter is stored in an object type document, which is a type of object for storing text and formatting information. This document object includes a link to a separate object of type "image". The display is accomplished by a document object manager displaying the text and the image object manager displaying the picture. The information describing the link is communicated from the document object manager to the image object manager with the assistance of the operating system's Application Integration Services (col. 12, lines 60).

For most object types each object is stored in a separate file (col. 15, line 5). If a destination object is not capable of storing data from a source object then the data is encapsulated. For example, a document object cannot directly store picture data (col. 18, line 25). In order to edit encapsulated data, the user must select the data and issue an edit command, thereby invoking an object manager capable of handling objects of the type in which the encapsulated data is stored: because of this difference encapsulated data will typically be visually marked for the user (col. 18, lines 33-40). The user will observe that editing the picture requires an extra operation that results in opening a new window (col. 18, line

44). No changes occur in the children while the viewer is viewing the parent (col. 19, line 23).

Link markers are included in the body of an object's data to indicate the presence of linked data (col. 20, line 11). The link marker is used to identify the object type so that the child (source) object's object manager can be called to display the child (source) object's data (col. 20, line 25). A link marker need not be physically stored at the location in the parent's data where the linked data is to appear (col. 20, line 45). Furthermore, the link marker does not specify the location of encapsulated data. The source object's object manager is used to locate that object's encapsulated data (col.15, line 48).

3. Hansen.

Hansen describes utilizing Andrew as a programming environment for authoring and editing software programs which are made up of multiple visual sub-languages. The reference teaches the creation of hierarchically-embedded windows within a document which provide views and interfaces to sub-elements of the parent document as depicted in Fig. 1. The *Layout* command provides for scattering objects in a rectangle as depicted in Fig. 1 (page 258, second column, 4th para.). Hansen states that the author should have the power to organize a program's constituent fragments for perusal by the reader. Since Hansen's system is intended for use with visual languages, which are languages that employ various graphical symbols to represent program elements and relationships between program elements, Hansen points out that it is preferable for the author to graphically arrange the various program fragments within a single window, in order to aid reader comprehension of the program, as a whole. Hansen makes no reference to embedding of any external executable program objects within a document being edited by a program author. Indeed, Hansen makes no reference to external executable program objects at all.

THE EXAMINER'S REASONING

The Examiner states that the Applicants' prior art (Mosaic) discloses utilizing the browser to display a portion of a hypermedia document that includes an embed text format that specifies the location an object external to the hypermedia document. It is also stated that is apparent that type information is specified which is used by the browser to identify and locate an executable application external to the hypermedia document.

With regard to Khoyi, the Examiner states that Khoyi teaches an object data processing system in which an extensible set of objects may be embedded within an object. The system invokes a corresponding object manager (a program external to the document) in response to an invocation request to process and control the object and teaches links specifying the object and type.

The Examiner acknowledges that Mosaic and Khoyi disclose launching the external application into a separate window to process the object.

However, the Examiner states that Hansen teaches that displaying each sublanguage element in a separate window runs the risk of chaotic imagery among which is difficult to discern the relationships among program segments.

The Examiner concludes that it "would be obvious to combine Khoyi with the Applicants' prior art to improve the system by providing open-ended for integrating new object/application and new type of information without recompiling the browser" (col. 12, line 65 to col. 13, line 28). The Examiner also concludes that, based on the Hansen disclosure, it would have been obvious to provide an external application to display and process the object within the browser-controlled window because it would have improved the system by reducing clustering of the display and aiding the reader comprehension of the hypermedia document.

TRAVERSE

The above rejection is respectfully traversed. The traverse is organized into two parts. Part I establishes that the features recited in claim 1 are not disclosed in the cited references. Part II establishes that: 1) there is no suggestion in the prior art that would make the claimed invention obvious; 2) the exercise of invention would be required by one skilled in the art, apprised of the teachings of the cited references, to make the claimed combination; 3) the commercial success of products, developed subsequent to filing the present application, incorporating the claimed features establishes that the combination was not obvious at the time of invention.

Part I

A. Mosaic does not disclose the recited embed text format that is parsed by the browser to initiate processing to automatically invoke an executable application external to the hypermedia document.

As described above, in Mosaic the URL is an address to an object. A URL anchor in Mosaic is not an embed text format that is parsed by the browser to initiate invocation of an executable application external to the document. Rather, when the anchor is activated, by the user interactively selecting the anchor, the browser retrieves the object and, if the object is another hypermedia document, replaces the first document with the second document. If the object has a file name associated with a helper application the application is launched and the object is viewed and/or edited in a separate window controlled by the helper application.

Accordingly, the external application is not automatically invoked as a result of the browser parsing the hypermedia document text, as required by the claim, but rather it is invoked by an interactive command given by the user, namely interactively selecting the URL anchor.

Further, a display window is not created in the first hypermedia document at the location in the document of the embed text format as required by the claim.

B. Khoyi et el.

This reference discloses an operating system based on capabilities similar to OLE, as used for example in Windows 95, as described at page 5, lines 27-33 of the present application. Source data in a form not displayable by a document manger is displayed in a window by an object manager which can display the source data. In the example described above, a picture is displayed in a document by the Khoyi system.

However, the display of the source data in the destination object is non-interactive. As stated above, the source data displayed in the destination document is delimited to be recognizable to the user. If the data is to be processed it must be interactively selected by the user and the source object manager invoked to process the data in a separate window. Depending on the link, the updated data will be displayed when the source object manager is displayed.

Thus, the presentation of the source data in the destination document is non-interactive. In the claimed invention, the external object is displayed in a window in the document and interactively processed using the external executable application. As set forth in the attached Doyle declaration, the claimed invention "lifted the glass" of the browser display to allow interactive control of document elements while being displayed in the browser controlled window. The Applicants' claimed invention allowed these elements to become "active" or "live" without requiring external programs to be first launched by the user's interactive commands. Furthermore, the Applicants' invention accomplished this functionality without requiring Khoyi-like capabilities in the operating system, making it practical for widespread use on a variety of operating systems. The claimed invention is a quantum leap over the disclosure of Khoyi or other OLE-type operating systems.

C. Hansen et al.

This reference discloses a programmer's source-code editing environment for visual languages that allows sub-elements

of the program being edited to be displayed in hierarchically-embedded subdocument windows. The Hansen system does not teach embedding, within the source-code document, of any executable applications which are external to the source-code document being edited.

Part II

1) There is no suggestion in the prior art that would make the claimed invention obvious.

Applicants' invention solves a different problem than each of the references, and such different problem is recited in the claims. In re Wright, 6 USPQ 2d 1959 (1988).

The Applicants' invention allows the hypermedia document to act as a coordinator and deployment mechanism, as well as a container, for any arbitrary number of external ***interactive*** data/application objects, while hiding the details of such coordination and deployment from the document's reader as the reader uses the various data/application objects. This allows the hypermedia document to act as a platform for entirely new kinds of applications that could not have been possible before the invention.

Because most of the functionality exposed to the reader is defined most directly by the hypermedia document, rather than any specific computer operating system, document-based applications using the invention tend to have the same look and feel to the reader, regardless of what type of computer or operating system is being used to run the browser application.

Additionally, because the embed text formats in the document cause the browser to automatically invoke the external application the document, the hypermedia document itself, and by implication the author of that document, directly control the extension of the functions of the browser.

Mosaic displays links, embedded in a first hypermedia document, and retrieves information identified by a link when a user activates the link. The retrieved information either replaces the first hypermedia document, or is displayed in a

separate window than the window displaying the first hypermedia document. Mosaic has the capability of invoking external applications to open a new window to display file types that cannot be displayed by Mosaic (helper apps).

Mosaic was a significant advance that made the WWW accessible and gave document authors a powerful tool to provide references external objects anywhere on the WWW.

The Khoyi system is an object oriented operating system that allows different types of objects to be integrated and object managers (applications) developed to manipulate and display the objects. The operating system coordinates interaction between the object managers, but has no knowledge of how any particular object is handled. An important advantage of the system is that new types of object-handling capability can be added without modifying the existing code. Thus the system is extensible.

OLE-style linking of objects is enabled by allowing different types of object data to be displayed in one document. The object managers for the different types of data are coordinated by the operating system so that each type of displayed data is rendered by its associated object manager. The actual linking operations are coordinated by the operating system.

Khoyi is an advance that allows applications to display different types of data and work together seamlessly. By the use of links, when an object is displayed the linked data is displayed in its latest format.

However, as described in detail above, there is no provision, suggestion, or motivation in Khoyi to provide for interactive processing of source data actually being displayed in a destination document. There is no need for such a feature in Khoyi because the different object managers can be invoked by the user at any time and source data can be processed in a window opened for that purpose.

The Examiner reasons that it would have been obvious to combine the teaching of Khoyi with Mosaic to improve Mosaic by providing an open ended mechanism for integrating new

object/applications and new types of information without recompiling Mosaic.

It is respectfully asserted that this reasoning is incorrect for the following reasons. In Khoyi the operating system is extensible in the sense that new types of data may be displayed, for example, in a destination document without recoding the destination document display application. However, the functionality of the document display application is not extended. There is still no capability for interactively processing the source data within the destination document window while the destination document is displayed within the same window.

Thus, there is no suggestion in Khoyi of modifying Mosaic so that an external application, by analogy to Khoyi the source document manager, is invoked to display and interactively process the object within the document window while the document is displayed by Mosaic in the same window.

The Examiner relies upon Hansen's teaching that, in a programmer's source code editor, a programmer should have the power to organize various fragments of the program for perusal by a reader, in order to aid the reader in comprehending the program [p256 col.1 1st paragraph]. The Examiner then states: "Hence, it would have been obvious for one of ordinary skill in the art to provide external application to display and process the object within the browser-controlled window because it would have improved the system by reducing the display and aiding the reader comprehension of the hypermedia document."

However, there is no suggestion in Hansen that an executable application external to the programming editor environment should be displayed and interactively processed within a document window. There is no discussion of external application programs at all. The fact that Hansen teaches that it is good to graphically organize the sub-elements of a document for better comprehension would not suggest to the person of skill in the art to combine parts of one reference, Khoyi's object data processing system, with another reference in order to meet Applicants' novel claimed combination.

In view of the above, it is concluded that the cited references neither explicitly nor implicitly suggest the claimed combination to a person of ordinary skill in the art.

2) The exercise of invention would be required by one skilled in the art, apprised of the teachings of the cited references, to make the claimed combination.

The combination of Mosaic and Khoyi would require either that a) Mosaic be modified in view of the teachings of the present invention or b) Khoyi be modified in view of the teachings of Mosaic to make the claimed combination. The only two possible ways to attempt to combine Mosaic and Khoyi include either adding the functionality of Khoyi to Mosaic or implementing the functionality of Mosaic within the Khoyi operating system.

Turning first to modifying Mosaic, to combine these references as proposed would have required novel and unobvious inventive steps. One must first consider that Mosaic is an application program which operates on any one of three operating systems: UNIX, Windows, and the Mac OS. Much of the current commercial success of the World Wide Web is due to this cross-platform compatibility of Web browsers. The system taught by Khoyi, on the other hand, is *a fully-independent and proprietary operating system*. As is stated in section 1.5 of Khoyi, "The operating system of the present invention differs from the traditional operating system in that, firstly, the actual functions and services performed by the operating system are reduced to the minimum ... functions and services which would normally be performed by an operating system, together with many functions and operations which would normally be performed by the applications programs themselves, are performed by libraries of routines [pack routines]. Examples of services and functions performed by pack routines include, but are not limited to, input/output operations, graphics/text and display operations, file access and management operations, and mathematical operations."

As is shown in the enclosed Doyle declaration, the only ***presently-known*** way to combine an operating system with an application program which runs on a different operating system is through the use of what is presently-known as a "***virtual machine***." This concept of a virtual machine was invented by a team of engineers working at Sun Microsystems Inc., in 1995, and was first exposed to the world in the form of a Web-browser plug-in called the "Java Virtual Machine." ***This virtual machine technology was not known at the time of the Applicants' invention***. Therefore, any combination of the references attempted at the time of the Applicants' invention would have yielded an inoperable result.

As is further shown in the enclosed Doyle declaration, the Java Virtual Machine concept has, together with Web browser plug-ins and applets, been hailed by the Industry as inventive and innovative. Since use of such an inventive step would be required in order to combine Mosaic and Khoyi, it follows that such a combination would have required novel and nonobvious combinative steps not taught in the prior art.

Therefore, adding the functionality of Khoyi to Mosaic would have been impossible at the time of the Applicants' invention without the creation of new novel and nonobvious technology. Even if such a combination had been possible and operable at the time of the Applicants' invention, Mosaic would have had to be significantly modified in a number of additional complex and nonobvious ways to achieve the combination.

First of all, Mosaic would have had to have been modified to incorporate the elements of a "virtual machine," that is, a program, which runs on a first computer and operating system, that emulates all of the necessary operations and resources of a second computer, and is under the control of an application executing on the first operating system. Such operations and resources include, in part, the machine instructions, graphics and I/O devices, and file system of the second computer. This "virtual" second computer would have to possess all of the characteristics necessary to allow the execution of the operating system taught by Khoyi. The Khoyi

system would then have to be integrated with Mosaic so as to execute on this virtual machine under the control of Mosaic.

Next, some sort of data interchange interface would have to be constructed between Mosaic and the Khoyi virtual machine to allow Khoyi's "packs" to create data structures that could be transferred to Mosaic, and for messages created by Mosaic to be transferred to Khoyi.

Mosaic would then have to be modified to allow data object components of the document to be "linked" to Khoyi's applications which can process the data. These links would be defined by an external link table, and the linking relationship would not be affected by the text of the document.

These links would be distinguished from the HTML anchor links defined in the hypermedia document, which would require incorporating two incompatible linking systems to be maintained by the system. Mosaic teaches that a major advantage of the HTML document format is that all links should be defined by the document text. This teaches away from combining the two systems in the proposed way, since the result would be awkward, overly complex and difficult to maintain.

In order for Mosaic to display the results of any computations that may have been made on the data object during viewing of the hypermedia document, Mosaic would have to be modified to allow the Khoyi virtual machine to write data directly to the Mosaic document data structure.

Mosaic would also have to be modified to allow it to be caused to re-render the document window in response to any change in the object imposed by the external program. Such re-rendering would require synchronization messages to be continuously exchanged between the browser application and the external program. This would involve the creation of some kind of message event loop that would wait for re-rendering messages to come from Khoyi's external program.

Similarly Khoyi would have to be modified to synchronize with Mosaic so that changes to the data would cause to external program to send a message to Mosaic to cause it to re-render its display. *Of course, even after doing all of the*

above, the external applications still could not be interacted with from within the Mosaic document, as required by the claimed invention, since Khoyi must launch any external application into a separate window before the reader can interactively control it.

Turning second to modifying Khoyi, any implementation of the functionality of Mosaic in the Khoyi operating system would be inoperable, due to the incompatibilities between the two systems' linking systems described below, and due to the differences between the two systems in storing, locating and processing data objects. Furthermore, such a combination would be impractical and nonobvious since the resulting combination would not conform to popular hypermedia protocol standards and would not be operable on any of the three most popular operating systems in the industry: UNIX, Mac OS, and Microsoft Windows.

The two linking systems could not be combined into one, due to the architecture of Khoyi's object system. Since Khoyi does not represent links within the document itself, but rather uses a link table which is external to the document, some sort of mechanism would have to be created to allow such links to be fully defined by the document text itself. This modification would render Khoyi's object system inoperable, since Khoyi's entire application architecture depends upon the link tables being the source of all link definitions, and being accessible to all of the various programs that may have a need to operate on a given data object.

Furthermore, since a document in the Khoyi system does not allow the document author to explicitly define or control the definition of the link's internal details, the document itself cannot specify such details as the precise location of a data file on a remote network disk drive. The Web, on the other hand, employs a uniform resource locator (URL) construct to manage both link definition and object localization on networked systems, from within the Web document, under the precise control of the Web document author. The URL mechanism would be incompatible with the linking mechanism requirements imposed by the Khoyi operating system. Since the URL-based mechanism for linking and object management is one of the major requirements for a successful Web

browser, such an incompatibility would render the resulting system useless for its intended purpose. *Furthermore, even if the above combination was operable, the external applications still could not be interacted with from within the hypermedia document, as required by the claimed invention, since both Mosaic and Khoyi must launch any external application into a separate window before the reader can interactively control it.*

None of the modifications listed above are taught in the prior art. **Nor are there any suggestions in the prior art that the references should be combined.** *To combine Mosaic and Khoyi in the manner suggested would require a multiplicity of separate, novel, inventive and awkward combinative steps that are too complicated to be considered obvious.*

Combining Hansen with any combination of Mosaic and Khoyi, while perhaps possible, would produce features that are irrelevant to the present application. Such a combination would involve modifying the hypermedia document data structure to allow multiple hierarchical subdocument windows to be contained within a parent document. This would involve substantial modifications to the Mosaic document rendering engine, as well as the development of a new version of the HTML document definition protocol to allow definition of hierarchical relationships within subdocument elements. Such a protocol would be exceedingly complex and would likely be incompatible with existing HTML standards. Combining Hansen with Khoyi would involve novel and unobvious steps similar to those described above for combining Mosaic and Khoyi. *Furthermore, the features that would result from the combination of Hansen and Mosaic are irrelevant to the Applicants' claimed invention, since, even after the combination, they would not show external executable applications being embedded within Hansen's documents.*

Thus, the Applicants submit that combining Mosaic, Khoyi and Hansen would require novel and unobvious inventive steps, and that the Applicants' invention is therefore novel and unobvious.

3) The commercial success of products, developed subsequent to filing the present application, incorporating the claimed features establishes that the combination was not obvious at the time of invention

As is shown in the enclosed Doyle declaration, several major competitors have incorporated the features of Applicants' invention into products, rather than to use the techniques of the prior art. The most notable of these products include the Navigator Web browser application from Netscape corporation, the ActiveX applet system from Microsoft corporation, and the Java Web applet system from Sun Microsystems corporation. The enclosed Doyle declaration further shows that the success of these products is directly attributable to the features of the claimed invention which each of these products incorporate, including an embed text format that is parsed by a Web browser to automatically invoke an external executable application to execute on the client workstation in order to display an external object and enable interactive processing of that object within a display window created at the embed text format's location within the hypermedia document being displayed in the browser-controlled window.

Some of these competitors have made laudatory statements about the elements of the Applicants' invention which are incorporated into their respective products, and have characterized those features as being a significant advance over prior art techniques.

Products incorporating the features of the invention have attained extensive commercial success

It is well known that, in the 1966 case of Graham v. John Deere, the U.S. Supreme Court decreed that Section 103 is to be interpreted by taking into consideration ***"secondary and objective factors such as commercial success, long-felt but unsolved need, and failure of others."***

As is shown in the enclosed Doyle declaration, there is universal acceptance within the computer software industry that the aforementioned products incorporating claimed features of the Applicants' invention have attained an extremely high degree of commercial success. Java, Navigator and ActiveX (all products incorporating features of the invention) represent among the most popular current technology in the computer industry for new application development.

The vast degree of commercial success that products incorporating features of the Applicants' invention have attained argues strongly ***against*** obviousness of the invention, and strongly ***for*** the patentability of the Applicants' claims.

The products incorporating features of the invention by others have been given many awards and have received considerable recognition in professional publications.

As is shown below, and in the enclosed Doyle declaration, Netscape, ActiveX and Java, all incorporating features of the Applicants' invention, have each been lauded as among the most innovative technologies to appear in the computer industry in recent years.

Some examples are:

The 1996 Discover Awards for Technical Innovation -- to Java and Navigator

PC Magazine 1995 Technology of the Year Awards -- To Java and Navigator

New Media Magazine 1996 Hyper Awards -- to Java and Navigator

New Media Magazine 1997 Hyper Awards -- to Active

As is evidenced in the enclosed Doyle declaration, this acclaim is due to the innovative nature of features of the claimed invention incorporated into those products and argues strongly against the obviousness of the Applicants' claims and argues strongly for the patentability of those claims.

Accordingly, since the Applicants' Claim 1 defines novel and unobvious structure that provides new and unexpected results as described above, and also because of the other numerous arguments

against the obviousness of Applicants' invention, made above, Applicants submit that Claim 1 is clearly patentable.

THE REJECTION OF CLAIMS 1-5

The Dependent Claims are A Fortiori and Independently Patentable over Mosaic, Khoyi and Hansen

Amended dependent claims 2 to 5 incorporate all the subject matter of Claim 1 and are therefore patentable for the same reasons as claim 1. Further, claims 2-5 add additional subject matter which makes them further and independently patentable over these references.

Claims 2-5 are rejected under 35 U.S.C. §103 as being unpatentable over Applicant's disclosed prior art, Khoyi, Hansen, and further in view of Moran "Tele-Nicer-Dicer: A new tool for the visualization of large volumetric data".

The rejection of Claim 2 on Mosaic, Khoyi, Hansen and Moran is overcome

Applicants' Claim 2 recites the additional step over Claim 1 of interactively controlling the controllable application on the client workstation via inter-process communications between the browser and said controllable application.

The disclosure of Mosaic, Khoyi, and Hansen has been described above. The reference Moran discloses a tool for interactive visualization of large, rectilinear volumentric data called Tele-Nicer-Slice-Dicer (TNSD). TNSD is based on client-server design where the client-side process is an extended version of a stand-alone visualization tool and the server process runs on a high-performance system where the data are located.

The client-side process describes data sets by text fields. Each data set description is used as a command which is sent to the server when a volume from the corresponding data set is requested. The use of a remote server is transparent.

The Examiner states that it would have been obvious to utilize the Moran application as an external application ("Viewer") in the prior art system as modified because it would

have improved the system by enabling the client station access to resources on higher performance serves to have interactive visualization of large data set capability.

Neither Mosaic, nor Khoyi, nor Hansen shows an executable application which is external to a document being displayed and interactively processed within that document's display window, nor do they show such an application where said executable application is interactively controlled on said client workstation by interprocess communications between the external application and the browser. This feature produces surprising and unexpected results over the prior art, since it allows the reader to perform all necessary interactive functions with external applications without directing his or her attention away from the hypermedia document. Additional surprising and unexpected results are yielded by the fact that the hypermedia browser application can have its functionality extended without making any changes to the hypermedia browser's object code. Further, surprising and unexpected results come from the ability of the document author to design interactive hypermedia document content that displays a similar look and feel to the reader, regardless of what the underlying operating system or computer platform the browser program is being executed upon.

The amendments to these claims have made the Moran reference irrelevant to the case, since Moran teaches a remotely-networked application being controlled via communications over a network, not an embedded (in a hypermedia document) interactive external application on the client workstation being controlled via inter-process communications between the document browser application and the external application.

Even if Moran was still in some way relevant, and even if the proposed combination was possible, was suggested by the prior art, and showed the features of the invention, all of which the above arguments for Claim 1 clearly show is not the case, the fact that a large number of references (more than 3) must be combined to meet the invention is further evidence of unobviousness

The rejection of Claim 5 on Mosaic, Khoyi, Hansen and Moran is overcome

Applicants' Claim 5 shows the additional steps over Claim 2 of communications to interactively control said controllable application which continue to be exchanged between said controllable application and said hypermedia browser even after said controllable application program has been launched.

None of the cited references show this feature. This feature produces the additional unexpected and surprising results over the prior art of allowing the browser application and the external application to precisely coordinate their activity, such as caching of the external application and shutting down its execution when no longer needed, entirely under the control of the browser application, in a manner that is transparent to the user. This drastically clarifies and simplifies the user's use of the hypermedia document and its related embedded applications.

The rejection of Claim 3 on Mosaic, Khoyi, Hansen and Moran is overcome

Applicants' Claim 3 shows the additional steps over Claim 5 of "additional instructions for controlling said controllable application reside on said network server, wherein said step of interactively controlling said controllable application includes the following sub-steps: issuing, from said client workstation, one or more commands to the network server; executing, on said network server, one or more instructions in response to said commands; sending information from said network server to said client workstation in response to said executed instructions; and processing said information at the client workstation to interactively control said controllable application."

None of the cited references show this feature. This feature leads to the additional surprising an unexpected results over the prior art of allowing the user to employ the hypermedia document as an interface to control and/or edit data objects which reside on the network server, remotely, from the client workstation. One of many possible uses of this feature is to

allow the user to make modifications to the original data object, which may remain in place on the network server, and which is referenced in the hypermedia document, so that others viewing the hypermedia document in the future from other client workstations will see those modifications.

The rejection of Claim 4 on Mosaic, Khoyi, Hansen and Moran is overcome

Applicants' Claim 4 shows the additional steps over Claim 3 "wherein said additional instructions for controlling said controllable application reside on said client workstation."

None of the claimed references show this feature. This feature produces the additional surprising and unexpected results of enabling a client & server system to be self-contained on the client workstation.

The rejection of claims 44-48 is overcome

Claims 44-48 are apparatus of the same scope as claims 1-5 and are thus allowable for the reasons recited above.

Accordingly Applicants submit that the dependent claims are a fortiori and independently patentable and should also be allowed.

Conclusion

For all of the above reasons, Applicants submit that the claims are now in proper form, and that the claims all define patentability over the prior art. Therefore they submit that this application is now in condition for allowance, which action they respectfully solicit.

Conditional Request for Constructive Assistance

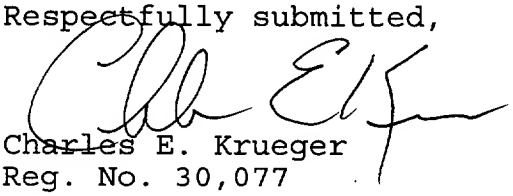
Applicants have amended the claims of this application so that they are proper, definite, and define novel structure which is also unobvious. If, for any reason, this application is not believed to be in full condition for allowance, Applicants respectfully request the constructive assistance and suggestions of the Examiner pursuant to M.P.E.P Section 706.03(d) and Section

707.07(j) in order that the undersigned can place this application in allowable condition as soon as possible and without the need for further proceedings.

In view of the foregoing, Applicants believe all claims now pending in this application are in condition for allowance. The issuance of a formal Notice of Allowance at an early date is respectfully requested.

If the Examiner believes a telephone conference would expedite prosecution of this application, please telephone the undersigned at (415) 576-0200.

Respectfully submitted,



Charles E. Krueger
Reg. No. 30,077

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
(415) 576-0200
Fax (415) 576-0300
CEK:db

i:\cek\share\02307i\553\june2.amd

#14
Party Paper 9

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231, on June 2, 1997

TOWNSEND and TOWNSEND and CREW LLP

By [signature]

PATENT

Attorney Docket No. 02307I-553

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:)
)
MICHAEL D. DOYLE et al.)
)
Application No.: 08/324,443)
)
Filed: 10/17/94)
)
For: EMBEDDED PROGRAM OBJECTS IN DISTRIBUTED HYPERMEDIA SYSTEMS)

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

I, MICHAEL D. DOYLE, hereby declare that:

1. I am a co-inventor of the subject matter disclosed and claimed in U.S. Patent Application No. 08/324,443.

2. I am an expert in the subject matter claimed in the referenced patent application as evidenced by the following curriculum vitae:

MICHAEL DAVID DOYLE

Born: April 27, 1959, in Chicago, Illinois

Education

- Ph.D. in Anatomy and Cell Biology, 1991, Department of Cell and Structural Biology, School of Life Sciences, University of Illinois at Urbana-Champaign
- Bachelor of Science, with honors, 1983, Department of Biocommunication Arts, University of Illinois at Chicago, Health Sciences Center

Employment

- Chairman and CEO, Eòlas Technologies Inc., Chicago, IL (http://www.eolas.com, email: mike@doyles.com)
- Adjunct Professor, 1996-present, Department of Computer Science, DePaul University of Chicago, Chicago, IL
- Asst. Adjunct Professor, 1994-present Department of Anatomy, School of Medicine, Univ. of Calif., San Francisco
- Director, Center for Knowledge Management, 1993 - 1994, (the UCSF Academic Computing and Informatics Research Center), University of California, San Francisco
- Director, UIC Biomedical Visualization Laboratory,1990 - 1993, Department of Biomedical Visualization, College of Associated Health Professions, University of Illinois at Chicago
- Assistant Professor, 1989 - 1993, Department of Biomedical Visualization, College of Associated Health Professions, University of Illinois at Chicago

Selected Professional Societies

Sigma Xi; Phi Kappa Phi; Mensa; International Society of Lymphology; Healthcare Information and Management Systems Society; Association for Computing Machinery, UIUC Chapter: Chairman and Founder, Special Interest Group for Biological

Computing, (SIGBIO), member ACM national; IEEE Engineering in Medicine & Biology Society; SPIE - The International Society for Optical Engineering

Selected Honors and Awards

Procter and Gamble Fellowship in Cell Biology, 1988-1989; National Research Service Award, Cell and Molecular Biology Training Program of the National Institutes of Health, University of Illinois at Urbana-Champaign, 1987-88; Nominated for the Charles A. Dana Award for Pioneering achievements in higher education, 1988; Francis and Harlie M. Clark Research Support Award, University of Illinois, 1987; University of Illinois Research Fellowship, 1987

Selected Committee and Board Memberships

- NIH Special Emphasis Panel: Computer Applications in Biology and Medicine, Division of Research Grants, National Institutes of Health, 1993-present
- Scientific Advisory Board: The Visible Human Project, National Library of Medicine, National Institutes of Health, 1991-1994
- Scientific Advisory Board: Center for Human Developmental Anatomy, National Museum of Health and Medicine, Armed Forces Institute of Pathology, 1992-1995
- Heads of Information Systems Committee, University of California System, 1993-1994
- Executive Committee: Integrated Advanced Information Management Systems, University of California, San Francisco, 1993-1994
- Organizational Committee for the First NIH Workshop on Osteoporosis Research in Dental Science, National Institute for Dental Research, 1991
- Curriculum Committee, College of Medicine, University of Illinois at Urbana-Champaign, 1986-1988

Selected Publications (of 54)

- Doyle, M.D.: *The Visible Embryo: Embedded Program Objects for Knowledge Access, Creation, and Management through the World Wide Web,* Computerized Medical Imaging and Graphics, 20/6(1996)
- Williams, B.S., and M.D. Doyle: *An Internet Atlas of Mouse Development,* Computerized Medical Imaging and Graphics, 20/6: 433-447 (1996)
- Doyle, M.D, C.S. Ang and D.C. Martin: *Proposing a Standard Web API,* lead article in Dr. Dobb's Journal, February (1996)
- Doyle, M.D., C.S. Ang, and D.C. Martin: *Embedding interactive external program objects within open-distributed-hypermedia documents,* High Speed Networking and Multimedia '94, SPIE Press (1995)
- Ang, C.S., D.C. Martin and M.D. Doyle: *Integrated Control of Distributed Volume Visualization Through the World Wide Web.* Proc. Visualization '94, IEEE Press (1994)
- Doyle, M.D, C. Ang, R. Raju, G. Klein, B.S. Williams, T. DeFanti, A. Goshtasby, R. Grzesczuk, and A. Noe: *Processing cross-sectional image data for reconstruction of human developmental anatomy from museum specimens.* SIGBIO Newsletter (The Jnl. of the ACM SIG for Biological Computing), 13/1 (1993)
- Carlbom, I., W.M. Hsu, G. Klinker, R. Szelski, K. Waters, M.D. Doyle, J. Gettys, K.M. Harris, T.M. Levergood, R. Palmer, L. Palmer, M. Picart, D. Terzopoulos, D. Tonnessen, M. Vannier, and G. Wallace: *Modeling and Analysis of Empirical Data in Collaborative Environments,* Communications of the ACM, 33/6, 75-84 (1992)
- Doyle, M.D.: *The MetaMAP Process: A New Approach to the Creation of Object-oriented Image Databases for Medical Education,* Proc. 13th Annual International Conference of the IEEE Eng. in Medicine and Biology Society, IEEE Press (1991)

- Doyle, M.D.: *The Use of Fractal Analysis in the Screening of Medical/Dental X-ray and Tomographic Images for Early Signs of Osteoporosis,* Proceedings of the 13th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE Press (1991)
- Doyle, M.D.: *A New Method for Identifying Features of an Image on a Digital Video Display,* Biostereometrics Technology and Applications, SPIE Press (1991)
- Doyle, M.D.: *The Interactive Digital Video Interface Process,* "Spatial Displays and Spatial Instruments", NASA Conference Publication #10032 (1989)
- Doyle, M.D., P.J. O'Morchoe, V. Navas, and C.C.C. O'Morchoe: *A Combined Enzyme Histochemical and Computer Image Analysis Technique for the Identification of Intraorgan Lymphatic Vessels* in: Progress in Lymphology-XI, H. Partsch, editor, Elsevier Science Publishers (1988)

Patents

- Lead inventor of "*Embedded program objects in distributed hypermedia systems,*" together with two additional co-inventors. The US patent is currently pending approval (assigned to the University of California, 1994). This patent is for the original development of World Wide Web plug-in and applet technology in 1993. It covers such currently popular technologies as Java, Navigator plug-ins, and Virtual Reality Markup Language (VRML). The first public demonstration of this work was given by Dr. Doyle's group at the second meeting of the Bay Area SIGWEB (Special Interest Group for the World Wide Web), at Xerox PARC, in November, 1993.
- Sole inventor: Awarded United States, Canadian and Australian patents for the development of what may have been the earliest hypermedia imagemap technology, entitled "*Method and Apparatus for Identifying Features of an Image on a Video Display*" (US,

#4,847,604, issued on July 11, 1989). Currently has related patent pending in Japan. In addition to its use for Web imagemap servers, this technology is also currently in use throughout the computer game industry as a method for optimizing collision detection algorithms.

Selected Invited Colloquia (of 23)

- "*Interactive Content on the Web: The Next Wave of Computing,*" keynote address at the first Internet@Chicago conference, sponsored by the Chicago Software Association, May, 1996
- "*The Embryology Metacenter: Biological applications of supercomputing in a distributed hypermedia environment.*" sponsored by the Department of Anatomy and Cell Biology, School of Medicine, Univ. of Michigan, Ann Arbor, March, 1994
- "*The Visible Embryo Project: A National Resource for Biomedical Information Technology,*" presented at the Annual Meeting of the American Association for the Advancement of Science, San Francisco, CA, February, 1994
- "*Accessing the Knowledge Base of Medicine in a Networked Environment,*" UCSF Grand Rounds in Medical Education, University of California, San Francisco, October, 1993
- "*Televisualization for the Support of Research and Education in Developmental Anatomy,*" a course in the Telemedicine Seminar and Workshop, sponsored by the Armed Forces Institute of Pathology, American Registry of Pathology, and the International Academy of Telemedicine, May, 1993
- "*The Embryology Metacenter: A distributed computational resource 'center' for developmental anatomy,* Second Annual Conference on Human Developmental Anatomy, National Museum of Health and Medicine, Washington D.C., December, 1992

3. In the course of my work in both academic research and in the software idustry I have supervised and have been associated with designers and developers of software, and I have personal knowledge of the ordinary level of skill in the art relevant to the claimed invention.

4. In my opinion the combination of the Mosaic and the Khoyi and Hansen references would not make the subject matter of claim 1 obvious to a person of ordinary skill in the art. I base this opinion on my belief that, firstly, based on the state-of-the-art in software technology prior to October of 1994, the combination of Mosaic and Khoyi would have been impossible or nonfunctional without the invention of novel and nonobvious technology, and secondly, the combination of Mosaic and Khoyi, or of Hansen with these two other references, would not have shown the features of the applicants' invention.

The only two possible ways to combine Mosaic and Khoyi would have been to either a) modify Mosaic in view of Khoyi or b) modify Khoyi in view of Mosaic. The discussion below shows that either approach would have been infeasible.

a) Modifying Mosaic in view of Khoyi:

Mosaic is an application program that was available in 1994 for the UNIX, Mac and Windows platforms. The system disclosed by Khoyi was a fully-independent operating system, developed by Wang Laboratories for use on proprietary word processing systems. In 1994, there was no published work, that I know of, that taught any method which would allow an application program, such as a Web browser, developed for one operating system, to "embed" the functionality of a different operating system, such as Khoyi, while still maintaining compatibility with the original operating system that the Web browser was developed

for. Even in light of the technologies available today, in 1997, there is no obvious solution to such a problem.

If asked to attempt to accomplish such a combination today, a person with an ordinary level of skill in the relevant art would most likely attempt to employ a Java Virtual Machine (JVM), which would allow a simulated computer platform to be embedded within the Web browser. The JVM was first employed as a mechanism to insure cross-platform compatibility of applet programs embedded within Web pages. This technology was first demonstrated in 1995 by Sun Microsystems, and the virtual machine component of it is generally acnowledged to have been innovative and nonobvious (Appendix A, Reference #1). Although the bytecode interpreter element of the Java language derives from work done on Smalltalk in the 1970s (Appendix A, Reference #2), the virtual machine concept was absolutely new at the time of Java's release. As Reference #2 of Appendix A states, "The final requirement is what has stymied many attempts at ubiquity in the past. If you base your system on any assumptions about what is "beneath" the run-time system, you lose. If you depend in any way on the computer or operating system below, you lose. **Java solves this problem by *inventing* an abstract computer of its own and running on that."**

Prior to Java, simulated computers had already been developed, but the JVM within the HotJava Web browser was the first known instance of a simulated computer system being incorporated *into* an application program running on another computer system. This allowed the Web browser to control the execution of applications running on the virtual machine, and to exchange information with those applications.

Even though there is no evidence that the JVM is robust enough to support a complete operating system, such as disclosed by Khoyi, there is a possibility that such an implementation could be accomplished. Alternatively, a new virtual machine could possibly be designed specifically for this purpose.

Attempts are underway at companies such as IBM to develop so-called universal virtual machine technology that can run programs written in languages other than Java. As Reference #3 of Appendix A points out, "Once developed, a universal virtual machine would, in theory, allow developers to write an application in any language and run it on any system. But at this stage, the challenges associated with developing UVMs are immense."

Assuming that the JVM would suffice for the purposes of combining the references, a person with an ordinary level of skill in the relevant art would then attempt to use the Java programming language to recreate the functionality of the Khoyi operating system on top of the JVM platform. Assuming that it would be possible to replicate the Khoyi system in the Java language, It is not at all clear that the subsequent combination with Mosaic would be useable, since there are a number of incompatibilities between the fundamental architectures of World Wide Web document systems and the type of document linking and object management system described by Khoyi. These incompatibilities are described below, in the section on modifying Khoyi in view of Mosaic.

b) Modifying Khoyi in view of Mosaic:

In order to modify Khoyi in view of Mosaic, one would attempt to implement a new document type on the Khoyi operating system through the development of an object manager to process documents formatted in the Hypertext Markup Language (HTML). Such an implementation would have been nonfunctional due to fundamental incompatibilities between the document linking and object management architectures in the two systems.

These incompatibilities relate to the way that Web documents handle object location and retrieval, and hypermedia links between documents and objects. The Khoyi system uses operating system services in order to manage the definition and resolution of links between data objects. Each link has a unique

identifier, which is referenced in a document, for example, as a "link marker." The actual definition of the link referenced by any particular link marker is located in an operating system data structure called a "link table." The document itself does not allow the document author to explicitly define or control the definition of the link's internal details, such as the precise location of a data file on a disk drive. The Web, on the other hand, employs a uniform resourse locator (URL) construct to manage both link definition and object localization on networked systems, from within the Web document, under the precise control of the Web document author. It appears that the URL mechanism would be incompatible with the linking mechanism requirements imposed by the Khoyi operating system. Since the HTML-based mechanism for linking and object management is one of the major requirements for a successful Web browser, such an incompatibility would likely render the resulting system useless for its intended purpose.

An attempt to actually perform the proposed combination of Mosaic and Khoyi would likely reveal further incompatibilities, but the obvious linking and object management problems are sufficient to conclude that the combination of these references is infeasible without the development of new and unobvious technology.

This opinion is further supported by statements made in the press by Microsoft Corp. concerning the difference between their Khoyi-like Object Linking and Embedding (OLE) technology and their ActiveX technology, first released in 1996, which allows the implementation within HTML documents of the features of the claimed invention, namely an embed text format that is parsed by a Web browser to automatically invoke an external executable application to execute on the client workstation in order to display an external object and enable interactive processing of that object within a display window created at the embed tag's location within the hypermedia document being displayed in the

browser-controlled window. Although many believed ActiveX to be just "OLE for the Internet," the above discussion of the difficulty of combining Khoyi with Mosaic may partly explain the delay that Microsoft took in coming up with a competing technology to Sun's competing Java product. This is supported by Reference #14 from Appendix A (Web Techniques, 10/96), which recalls a conversation between Web Techniques' editor in chief and ActiveX product managers from Microsoft:

> "'[Web Techniques:]--So tell me about ActiveX. Isn't it just OLE for the Internet?'
>
> '[Microsoft:]--Wrong,... ActiveX is a new API that, like OLE, is based on Microsoft's Component Object Model (COM). While OLE supports a compound document architecture for desktops, ActiveX is designed specifically to embed rich media objects within Web-based documents.'
>
> This was part of a scene that recently played out in the offices of Web Techniques...Microsoft marketers arrived on our doorstep not for the standard dog-and-pony show, but specifically to 'debunk the myths' surrounding ActiveX technologies."

Of course, even if the combination of Mosaic and Khoyi had been possible and functional, it still would not have shown the features of the claimed invention, including an embed text format that is parsed by a hypermedia browser to automatically invoke an external executable application to execute on the client workstation in order to display an external object and enable interactive processing of that object within a display window created at the embed text format's location within the hypermedia document being displayed in the browser-controlled window. Furthermore, the additional combination of Hansen with these references would not have yeided the features of the claimed invention either. Hansen discloses a programmer's source-code editing environment for visual languages that allows sub-elements

of the program being edited to be displayed in hierarchically-embedded subdocument windows. The Hansen system does not teach embedding, within the source-code document, of any executable applications which are external to the source-code document being edited. Therefore the Hansen reference would not add features relevant to the claimed invention, even if combined with Mosaic and Khoyi.

The claimed invention lifted the glass, so to speak, from the Web browser, making it possible to embed fully-interactive external applications within Web pages, thereby turning the browser into a *platform* for the development of *entirely new* kinds of applications.

5. Further, in my opinion secondary considerations, including, in part, commercial success of products incorporating features of the claimed invention and industry recognition of the innovative nature of these products, demonstrate that the claimed invention is not obvious over the cited references.

The three exemplary products which incorporate the features of the claimed invention include Netscape Navigator 2.0 (or newer versions), Java, from Sun Microsystems, and ActiveX, from Microsoft. One need only open the pages of any major business publication to see that these three products have been tremendously successful in the marketplace. Appendix A of this declaration presents a collection of excerpts from prestigious Industry publications which support the contention that the success of these products is directly attributable to the claimed features of the invention.

Approximately 12 to 18 months after the applicants initially demonstrated the first Web plug-in and applet technology to the founders of Netscape and engineers employed by Sun Microsystems in November and December of 1993, as described in reference #4 from Appendix A (Dr. Dobb's Journal, 2/96), both Netscape and Sun released software products that incorporated

features of the claimed invention, **including an embed text format that is parsed by a Web browser to automatically invoke an external executable application to execute on the client workstation in order to display an external object and enable interactive processing of that object within a display window created at the embed text format's location within the hypermedia document being displayed in the browser-controlled window.** Sun released the Java applet programming environment and the HotJava applet-capable Web browser in May of 1995, and Netscape release version 2.0 of their Navigator Web browser, which incorporated both Java technology and a plug-in API, in October of 1995.

From Marc Andreessen's comments, quoted in Reference #5 of Appendix A (Wired, 12/95), it is clear that corporate management at Netscape put considerable emphasis on the "platform" nature of the browser (enabled by plug-in and Java technology which first appeared in version 2.0 of Navigator) in their marketing communications for the product. Andreessen stresses the term "live online applications" in describing the product's features. He says, "We use the term *live online applications* for the types of applications that people build on our platform - *online* because the applications are network-centric and dustributed, ***live* because they're highly interactive** with users and with data retrieved in real time from databases and other sources over the network. *Live Objects* is our term for things like Java applets and inlined viewers embedded in HTML documents. So, a live online application is built using HTML as a framework."

Andreessen's comments relate directly to the capabilities given to their Web browser through the use of "plug-in" applications and Java applets. These capabilities result directly from both the plug-ins' and Java's incorporation of the features of the claimed invention, including an embed text format that is parsed by a Web browser to automatically invoke an external executable application to execute on the client

workstation in order to display an external object and enable interactive processing of that object within a display window created at the embed tag's (or "applet" tag's) location within the hypermedia document being displayed in the browser-controlled window. Andreessen's use of the terms "live objects" and "live online applications" refer specifically to the seemingly "live" nature of these fully-interactive embedded windows within plug-in-enabled or Java-enabled Web pages, as opposed to the static nature of Web document content seen prior to the applicants' invention.

A good indicator that Sun Microsystems felt that enabling interactivity in Web pages was the key feature of Java is given in the first chapter of "Hooked on Java," which was written by members of the original Java development team. They say, "With applets written in the Java programming language, Web users can design Web pages that include animation, graphics, games, and other special effects. **Most important, Java applets can make Web pages highly interactive.**"

This statement shows that the developers of Java felt that the most important feature of the Java technology was the ability of Java to allow an embed text format (the applet tag) within a Web document to be parsed by a Web browser to automatically invoke an external executable application to execute on the client workstation in order to display an external object and enable interactive processing of that object within a display window created at the applet tag's location within the hypermedia document being displayed in the browser-controlled window. The book's authors further emphasize the novelty and nonobviousness of this technology when they say, "Quite simply, Java-powered pages are Web pages that have Java applets embedded in them. They are also the Web pages with the coolest special effects around....Remember, **you need a Java-compatible Web browser such as HotJava to view and hear these pages and to**

interact with them; otherwise, all you'll access is static Web pages minus the special effects."

The novel and nonobvious nature of the claimed invention is further emphasized by a quote taken from Reference #7 of Appendix A, where Marc Andresseen's early opinion of Java is reported: "In December 1994, Java and HotJava (at this stage called Oak) were posted in a secret file deep in the Net; only a select few were given pointers and invited to check it out. **Three months later, Marc Andreessen gushed to the *San Jose Mercury News* 'What these guys are doing is undeniably, absolutely new. It's great stuff.'"**

Others in the Industry supported this opinion as well, as Reference #8 clearly shows: "'The Java programming language radically advances the multimedia potential of the Net, enabling faster animation, games, and powerful interfaces within Web sites,' says Nova Spivack, director of marketing and co-founder, EarthWeb. **'Sun's Java technology is a stroke of genius** that will transform the Internet in a matter of months.' ... **'The Java language is a revolutionary technology** with profound implications for the Internet as well as the computer industry in general,' says Jack D. Hidary, chief executive officer, and co-founder of EarthWeb."

That the interactive nature of embedded interactive program objects was the specific reason for the commercial success of Java was forcefully illustrated in Reference #9 (Communications Week, 10/95): "The impact of Java on our [financial services] marketplace is that **static browsers have to become interactive and event-oriented,**" said Bill Adiletta, president at Market Vision, in Santa Cruz, Calif. **'That is absolutely essential for us to even consider [the Web] as an option for our marketplace.'"** The "interactive and event-oriented" capabilities which Mr. Adiletta refers to are those that are specifically enabled through the Web browsers'

incorporation of the features of the claimed invention, including an embed text format that is parsed by a Web browser to automatically invoke an external executable application to execute on the client workstation in order to display an external object and enable interactive processing of that object within a display window created at the embed tag's location within the hypermedia document being displayed in the browser-controlled window.

The status of Java, and by association, Netscape Navigator, as one of the most popular development platforms for new applications is supported by Reference #10 (Forbes, 11/96), which states that "**Java is also well on its way to becoming the most important Internet software standard,** catapulting Sun past Netscape and Microsoft as the leader in Internet computing."

That the results of the features of the claimed invention are surprising and unexpected is supported by the above references, and by Reference #10, which quotes Sun's CEO, Scott McNealy as saying, "We always thought we were onto something with Java--that it was our one big chance to challenge Microsoft and to change the economics of the business... But I have to admit I never thought Java computing could unfold quite this quickly, or with such universal support from customers and competitors. **It's astounding, really.**"

The novel and unobvious nature of the claimed invention is strongly supported by Reference #11 (PC WEEK, 4/96), which reports the award, by PC Week, of a technical innovation award to Java. They go on to state that "**The best example so far of why Java is more than a cool concept is Sun's Java WorkShop,** a development tool for creating applets that is completely written in Java. It is basically a set of Java applets that run on any platform that has a Java virtual machine, which for now includes Solaris and 32-bit Windows operating systems. A Macintosh version is due this summer. Java WorkShop has another unique quality that is likely to affect the way all applications are

developed in the future. **The WorkShop applets and the applications created with the development environment live within a Web browser** written in Java. The basic notion is that if you are developing for the Web, the containter should be Web-centric. **The user interface is a Web browser that integrates Net technologies like Hypertext Markup Language at the core of the product.** You click on an icon to load a new tool that lives on a local Web page, and you can go to other pages to access other tools, code and documentation." This award illustrates that there was Industry consensus that the most important part of the Java technology was its ability to allow external interactive applications to be embedded within Web pages.

Sun's Java WorkShop, as described in this reference, shows many of the features of the claimed invention. It is an excellent example of the surprising and unexpected results of the claimed invention.

Despite the advantages and popularity of the Java applet platform, it is not all things to all people. Web plug-ins, at least those that don't depend on applet interpreters, are oftentimes a more pratical solution for enhancing the interactivity of Web pages. Reference #12 (Boardwatch, 6/96) describes the feelings of one well-known expert in the field: "Well, even though Java is supposed to be the elixer for the woes of Internet standardization, every day seems to bring a new flavor - smoother, tastier, more powerful. How about sticking to one flavor and form so that someone can actually figure out what to do with it? On to the stuff that's real. **Plug-ins are currently the most useful browser enhancement.** The variety and quantity of plug-ins that have become available in the last few months has astounded me. They open up immense possibilities of all kinds for unique and interesting applications." Again, one should note that the term "plug-ins" used here specifically refers to programs which incorporate the features of the claimed invention, including an embed text format that is parsed by a Web

browser to automatically invoke an external executable application to execute on the client workstation in order to display an external object and enable interactive processing of that object within a display window created at the embed tag's location within the hypermedia document being displayed in the browser-controlled window.

Of course, Microsoft did not sit idly by during this period of rapid technological advance. Marc Andreessen comments on Microsoft's recent licensing of Mosaic in his Dec.'95 Wired interview (Reference #5): "We have a wide range of competitors entering this space, competing with one part of what we do. The Microsoft browser is basically what we did with Mosaic - I'm glad to see that they've caught up to what we did two years ago."

It is clear from this comment that Microsoft's browser at the time did not support embedded interactive program objects. It took them almost a year to come up with a competing technology, which they called ActiveX, and which was released in 1996.

Microsoft felt that embedded interactive program objects in Web pages (a feature of the claimed invention) was so fundamental a requirement for commercial success, that they even licensed Java for incorporation in their Web browser, at a time when it was apparent that doing so might be at Microsoft's own peril (from Reference #10, Forbes, 11/96): "Gates can thank little Netscape for putting the $9 billion software behemoth in the extraordinary position of having to support a technology that could badly undermine Windows. **Once Gates decided he wanted to unseat Netscape as the Internet browser king, Microsoft had to incorporate Java into its own browser, Internet Explorer, just as Netscape had done with Navigator....**Microsoft still derides Java as merely a 'mildly interesting programming language' and is doing all it can to torpedo Java with its own Internet software component technology, ActiveX. Microsoft claims ActiveX uses PC hardware and software better than Java does."

Like Netscape's Navigator and like Sun's Java products, Microsoft's ActiveX technology has become very successful. This is evidenced by the following excerpt from New Media Magazine's 1997 "Hyper Awards" issue, where they give ActiveX the award of "Technology of the Year":

"TECHNOLOGIES OF THE YEAR: Microsoft ActiveX

Microsoft ActiveX was first maligned as an unnecessary Java competitor and part of Bill Gates' plot to dominate the world, but this standard for Internet applets has assumed the high ground as a universal OLE-based wrapper that embraces Java, VRML, Shockwave, Visual Basic, C++, and other scripting tools.

To further establish it as an industry norm, Microsoft is ensuring development of **ActiveX plug-ins for Mac and UNIX Web browsers**, and is even spinning off control of ActiveX into an independent standards body, a first for Microsoft. The strategy is working -- ActiveX acceptance among developers has been phenomenal, and nobody is comparing it to Java anymore."

The above citations, as well as the additional details given in Appendix A, provide ample evidence of the commercial success of products incorporating features of the claimed invention, as well as evidence of the widespread acclaim that these products have garnered for the technical innovations which the features of the claimed invention allowed them to provide. They further show that the successes of these products was a direct result of the features of the claimed invention, which they incorporated *through implementation of an embed text format that is parsed by a Web browser to automatically invoke an external executable application to execute on the client workstation in order to display an external object and enable interactive processing of that object within a display window created at the embed text format's location within the hypermedia document being displayed in the browser-controlled window.*

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Dated: May 27th, 1997.

MICHAEL D. DOYLE

CEK:db
s:\02307I\553\DECL.01

Appendix A to Doyle Declaration

List of relevant references:

1) "Today the Web, Tomorrow the World," by Tom Halfhill, Byte, Jan. 1997:

> "Never in the history of computing has a new language attracted so much support from toolmakers, software developers, and OS vendors in such a short time.....**Java trancends being a language to being a software platform because of the Java virtual machine (VM), which simulates a computer in software.** The Java VM can run on existing computers and OSes (e.g., Windows and the Mac OS), or it can run on hardware designed only for Java....Java could trigger the biggest platform shift since Windows surpassed DOS--all without forcing you to change your hardware and OS...Java is a stealth platform that propagates entirely in software and coexists peacefully with the native OS....Java carries software abstraction to the next level because it abstracts everything below the VM. It's designed for a world in which the OS and CPU are interchangeable parts that can be replaced without breaking applications."

2) "Teach yourself Java in 21 days," by Laura LeMay and Charles Perkins, Sams Net Publishers, 1996, pages 423-424:

> "It [Java] deserves to be there [at center stage]. It is the natural outgrowth of ideas that, since the early 1970s inside the Smalltalk group at Xerox PARC, have lain relatively dormant in the mainstream. Smalltalk, in fact, invented the first object-oriented bytecode interpreter and pioneered many of the deep ideas that Java builds on today....The final requirement is what has stymied many attempts at ubiquity in the past. If you base your system on any assumptions about what is "beneath" the run-time system, you lose. If you depend in any way on the computer or operating system below, you lose. **Java solves this problem by *inventing* an abstract computer of its own and running on that."**

3) "Virtual Machines: Vendors move past Java to look for UVMs," By Jim Balderston and Bob Trott, InfoWorld, April 7, 1997:

> "IBM, in conjunction with its Taligent subsidiary, has begun a research effort intended to create a single virtual machine capable of running applications written in C++, Smalltalk, or Java....Once developed, a universal virtual machine would, in theory, allow developers to write an application in any language and run it on any system. But at this stage, the challenges associated with developing UVMs are immense, and any such mechanism would probably only be capable of running

interpreted code, which would have a big impact on performance and memory requirements."

4) "Proposing a Standard Web API," by Michael Doyle, Cheong Ang and David Martin, Dr. Dobb's Journal, February, 1996:

"We designed and implemented an API for embedded inline applets that allowed a Web page to act as a container document for a fully-interactive remote-visualization application, allowing real-time volume rendering and analysis of huge collections of 3-D biomedical volume data, where most of the computation was performed by powerful remote visualization engines. Using our enhanced version of Mosaic, later dubbed 'WebRouser,' a scientist using a low-end client workstation could exploit computational power far beyond anything that could ever be found in one location.

This work was shown to several groups in 1993, including many that were later involved in projects to add APIs and applets to Web browsers at places like NCSA, Netscape, and Sun."

5) "Why Bill Gates wants to be the next Marc Andreessen," Interview by Chip Bayers with Netscape founder, Marc Andreessen, Wired Magazine, December, 1995, page 165:

"**Bayers:** Because it allows plug-ins, you're calling Netscape 2.0 a *platform*, rather than a *browser* in the company's descriptions. But Photoshop has some pretty sophisticated plug-ins, and it's only an application. Is Netscape 2.0 a platform or an application?

Andreessen: It's a little of both. It's an application in that it runs on top of what is traditionally thought of as an operating system - like Windows or UNIX - but **it's a platform in that people can build applications on it.** We use the term *live online applications* for the types of applications that people build on our platform - *online* because the applications are network-centric and dustributed, ***live* because they're highly interactive** with users and with data retrieved in real time from databases and other sources over the network. *Live Objects* is our term for things like Java applets and inlined viewers embedded in HTML documents. So, a live online application is built using HTML as a framework. This gives developers great flexibility in linking together people and information over networks.

Our platform is also operating-system independent; you can run the client on Windows 3.1, Windows 95, Windows NT, UNIX - any of 12 flavors - or Mac; the server can run on Windows NT, UNIX, and in the near future, Windows 95.

We don't use the term *browser* because we think it's pretty clear that Netscape Navigator 2.0 is far more than a browser. On one hand it's a suite: it handles e-mail, threaded

discussion groups, ftp, gopher, chat, et cetera. On the other hand, **it's a platform: it allows people to build these live online applications on top of it."**

"**Bayers:** Since this is an entrepreneurial environment with a level playing field, who do you see as your biggest competitor right now? Microsoft and its new browser?

Andreessen: We have a wide range of competitors entering this space, competing with one part of what we do. The Microsoft browser is basically what we did with Mosaic - I'm glad to see that they've caught up to what we did two years ago."

6) "Hooked on Java," by Arthur Van Hoff, et al., of Sun Microsystems:

Page 1: "With applets written in the Java programming language, Web users can design Web pages that include animation, graphics, games, and other special effects. **Most important, Java applets can make Web pages highly interactive.** Of course, users need a Java-compliant Web browser like Netscape Navigator or HotJava to view and use Java-powered pages....Maybe we're a little bit biased because we are part of the programming team that has been developing Java at Sun, but we think it's true. Java allows you to do exciting things with Web pages that weren't possible before....Greater interactivity is one of the hallmarks of Web pages that use Java. Users can interact with the content of a Java-powered page via the mouse, keyboard, and other user-interface elements such as buttons, slides, and text fields."

Page 2: "Small programs written in the Java programming language called Java applets make this all possible. Java applets are embedded right in Web pages. When users access these pages, the applets are downloaded to their computers and executed. Instead of the activity happening on the server side as is the case with CGI programming, it happens on the client side in a Java-compatible Web browser."

Page 3: "Quite simply, Java-powered pages are Web pages that have Java applets embedded in them. They are also the Web pages with the coolest special effects around....Remember, you need a Java-compatible Web browser such as HotJava to view and hear these pages and to interact with them; otherwise, all you'll access is static Web pages minus the special effects."

7) "The Java Saga," by David Bank, Wired Magazine, Dec., 1995:

Page 166: "While today's Web is mostly a static brew - a grand collection of electronically linked brochures - Java holds the

promise of caffeinating the Web, supercharging it with interactive games and animations and thousands of application programs nobody's ever heard of....Software developers are busy shaping Java into applications that will add new life to Web browsers, like Netscape and Mosaic, producing programs that combine **real-time interactivity** with multimedia features that have been available only on CD-ROM...What's a Java application? Point to the Ford Motor web-site, for instance, and all you'll get are words and pictures of the latest cars and trucks. Using Java, however, Ford's server could relay a small application (called an applet) to a customer's computer. From there, the client could customize options on an F-seried pickup while calculating the monthly tab on various loan rates offered by a finance company or local bank."

Page 167: "Sun is giving away Java and HotJava for free for noncommercial use, in a fast-track attempt to make them the standard before Microsoft begins shipping a similar product, codenamed Blackbird, in early 1996" (note: Blackbird was later renamed to "ActiveX" by Microsoft)

Page 242: "A new business plan was drawn up early in 1994, which unceremoniously dumped the speculative markets FirstPerson had pursued and began focusing on personal computers - the technology the project was supposed to leapfrog in the first place. The new plan was to create a corps of CD-ROM developers who would write in Oak and, ideally, stick with it as their platform language while moving applications to the commercial online services....**The plan, remarkably, contained no mention of Mosaic or the Web**....FirstPerson was scrapped in the spring of 1994..."

Page 243: "Joy and Schmidt wrote yet another plan for Oak and sent Gosling and Naughton back to work adapting Oak for the Internet. Gosling, whom Joy calls 'the world's greatest programmer,' worked on the Oak code, while Naughton set out to develop a true 'killer app.'...In January 1995, Gosling's version of Oak was renamed the more marketable Java. Naughton's killer app was an interpreter for a Web browser, later named HotJava."

Page 244: "In December 1994, Java and HotJava (at this stage called Oak) were posted in a secret file deep in the Net; only a select few were given pointers and invited to check it out. **Three months later, Marc Andreessen gushed to the *San Jose Mercury News* 'What these guys are doing is undeniably, absolutely new. It's great stuff.'** That was how the Java team knew it was finally going to make it. 'That quote was a blessing from the god of the Internet,' Polese says"

Page 245: "Sun is racing to stay ahead of the accelerating wave. The day after that midnight deadline for sending the finished code to Netscape, Joy was already at work pushing the limits of what Java could do.... 'I've got 15 patents I could file as soon as I type them,' Joy says. 'I figure I've got five years. It's like we've got a blank sheet and it says 'Internet.' Normally the best products don't win. The Internet is an opportunity for the best products to win. Java is great.technically and people want it. I'm happy to get that once in my life - or maybe twice."

8) "Earthweb and Sun unveil Gamelan on the Internet; Premier directory for the Java revolution goes online," PRNewswire, October 11, 1995:

"'The Java programming language radically advances the multimedia potential of the Net, enabling faster animation, games, and powerful interfaces within Web sites,' says Nova Spivack, director of marketing and co-founder, EarthWeb. **'Sun's Java technology is a stroke of genius** that will transform the Internet in a matter of months.' ... **'The Java language is a revolutionary technology** with profound implications for the Internet as well as the computer industry in general,' says Jack D. Hidary, chief executive officer, and co-founder of EarthWeb."

9) "Goal/A multiplatform operating system/Sun positions Java as universal interface," by Richard Karpinski, Communications Week, October 9, 1995:

"What Sun sees with Java is the opportunity to build a universal, multiplatform operating system - a 'ubiquitous API,' said Schmidt - that will open up a new world of distributed, networked applications....According to Bill Joy, co-founder and chief technologist at Mountain View, Calif.-based Sun, **Java not only will enable more interactive and animated Web content,** but eventually it will replace C++ as the standard building block for computer applications, paving the way for a new paradigm of distributed, network-based applications....Nearer term, however, Java mainly will be used to jazz up the Web.... 'The impact of Java on our [financial services] marketplace is that **static browsers have to become interactive and event-oriented,'** said Bill Adiletta, president at Market Vision, in Santa Cruz, Calif. **'That is absolutely essential for us to even consider [the Web] as an option for our marketplace.'"**

10) "Sun's Java: The threat to Microsoft is real," by Brent Schlender and Eryn Brown, Forbes, November 11, 1996:

11) "The First Annual IT Excellence Awards," PCWEEK, April, 1996:

"We had planned to pick **one product that stood out from the pack in terms of innovation, advancement, and improvement in cost of ownership of information delivery technology. Java met those criteria....**[Java] may enhance Sun's role as a mainstream industry innovator and leader. Until recently, Java was famous as hypeware, exemplified by cute but shallow applets. As one of our readers said, it's 100% buzzword-compliant. Nonetheless, we give Java a PC WEEK Corporate IT Excellence award for innovation this week, and it appears that the language environment is quickly moving beyond novelty to practical application."

"The best example so far of why Java is more than a cool concept is Sun's Java WorkShop, a development tool for creating applets that is completely written in Java. It is basically a set of Java applets that run on any platform that has a Java virtual machine, which for now includes Solaris and 32-bit Windows operating systems. A Macintosh version is due this summer. Java WorkShop has another unique quality that is likely to affect the way all applications are developed in the future. **The WorkShop applets and the applications created with the development environment live within a Web browser** written in Java. The basic notion is that if you are developing for the Web, the containter should be Web-centric. **The user interface is a Web browser that integrates Net technologies like Hypertext Markup Language at the core of the product.** You click on an icon to load a new tool that lives on a local Web page, and you can go to other pages to access other tools, code and documentation."

12) "Browser Plug-Ins: Good, Bad and Ugly," by Chris Babb, Boardwatch, June, 1996:

"Well, even though Java is supposed to be the elixer for the woes of Internet standardization, every day seems to bring a new flavor - smoother, tastier, more powerful. How about sticking to one flavor and form so that someone can actually figure out what to do with it? On to the stuff that's real. **Plug-ins are currently the most useful browser enhancement.** The variety and quantity of plug-ins that have become available in the last few months has astounded me. They open up immense possibilities of all kinds for unique and interesting applications."

13) "1997 Hyper Awards," New Media Magazine, January, 1997:

"Originally known as a way to jazz up Web pages with graphic animations--stock tickers that crawl across your screen, for example, and dancing icons--Java has quickly evolved into a whole lot more. To Microsoft's dismay, it is fast becoming what is known as a computing platform--a sturdy base upon which programmers can build software applications. Java is making possible the rapid development of versatile programs for communicating and collaborating on the Internet....Java is also making possible a controversial new class of cheap machines called network computers, or NCs, which Sun, IBM, Oracle, Apple and others hope will proliferate in corporations and our homes....To graph the numbers, you'll call in a charting applet that will let you print out your report nice and pretty, **all without leaving your browser.** And you'll always get the latest, greatest version of the applets too: Since the software is stored in only one place, corporate techies can keep it up to date more easily....**Java is also well on its way to becoming the most important Internet software standard,** catapulting Sun past Netscape and Microsoft as the leader in Internet computing....**The halo effect from Java is a big reason Sun's stock has become hot.**"

"For Sun CEO Scott McNealy, it's a pipe dream come true. 'We always thought we were onto something with Java--that it was our one big chance to challenge Microsoft and to change the economics of the business,' he says, 'But I have to admit I never thought Java computing could unfold quite this quickly, or with such universal support from customers and competitors. It's astounding, really.' ... Java is also one of those charmed technologies--Microsoft's original DOS operating system is another--that arrived at exactly the right place at the right time. Since Sun introduced Java in May 1995, a constellation of forces--other Internet innovations, software economics, industry politics, and customer need--aligned almost simultaneously to let Java emerge....The keys to Java's success as a platform are ubiquity and absolute compatibility throughout the industry. To achieve ubiquity, Sun hitched a ride on Netscape's popular Navigator Internet browser, the program that unleashed the whole Internet phenomenon in the first place....Gates can thank little Netscape for putting the $9 billion software behemoth in the extraordinary position of having to support a technology that could badly undermine Windows. **Once Gates decided he wanted to unseat Netscape as the Internet browser king, Microsoft had to incorporate Java into its own browser, Internet Explorer, just as Netscape had done with Navigator....**Microsoft still derides Java as merely a 'mildly interesting programming language' and is doing all it can to torpedo Java with its own Internet software component technology, ActiveX. Microsoft claims ActiveX uses PC hardware and software better than Java does."

"TECHNOLOGIES OF THE YEAR:
Microsoft ActiveX

Microsoft ActiveX was first maligned as an unnecessary Java competitor and part of Bill Gates' plot to dominate the world, but this standard for Internet applets has assumed the high ground as a universal OLE-based wrapper that embraces JavaVRML, Shockwave, Visual Basic, C++, and other scripting tools.

To further establish it as an industry norm, Microsoft is ensuring development of **ActiveX plug-ins for Mac and UNIX Web browsers**, and is even spinning off control of ActiveX into an independent standards body, a first for Microsoft. The strategy is working -- ActiveX acceptance among developers has been phenomenal, and nobody is comparing it to Java anymore."

14) "ActiveX, a Standard?," by Michael Floyd (Editor in Chief), WebTechniques, October, 1996, page 5:

"'--So tell me about ActiveX. Isn't it just OLE for the Internet?'

'--Wrong,... ActiveX is a new API that, like OLE, is based on Microsoft's Component Object Model (COM). While OLE supports a compound document architecture for desktops, ActiveX is designed specifically to embed rich media objects within Web-based documents.'

This was part of a scene that recently played out in the offices of Web Techniques...Microsoft marketers arrived on our doorstep not for the standard dog-and-pony show, but specifically to 'debunk the myths' surrounding ActiveX technologies."